SEL – PW2

MAI

URV-UB-UPC

Author:

Alberto Soutullo Rendo

14-May-2021

Content

1. Introductions and selected datasets

The objective of this project is to implement, compare and validate two combinations of multiple classifiers: a **random forest** [1] and a **decision forest** [2]. The base-learner for inducing the trees is the CART method.

These two classifiers should be able to train and predict from a .csv file format. Also, they should produce an ordered list of features used in the forest according to its importance. Apart from this, an accuracy value should be also returned.

Regarding the hyperparameters, it is expected to use multiple values of number of random features (**F**) and number of trees (**NT**), like:

	Random Forest	***Decision Forest***
NT	1, 10, 25, 50, 75, 100	
F	1, 3, int($\log_2 M$ + 1), $\sqrt{M}$	Int(M/4), int(M/2), int(3*M/4), Runif*****

Runif is a function that generates pseudorandom integer values ru such as 1 <= ru <= M, being M the total number of features.

For this project, the division of the datasets into train and test was done with a **0.75** split for all the different evaluations. Also, the **same value** was used for bootstrapping the training set in the random forest classifier.

These two classifiers must be evaluated with at least 3 different datasets. The datasets must have different sizes, with less than **500**, between **500** and **2000**, and over **2000** instances.

The datasets selected for the evaluation are **glass** [3] **(214 instances, 10 attributes + class)**, **cmc** [3] **(1473 instances, 9 attributes + class)** and **kr-vs-kp** [3] **(3196 instances, 36 features + class)**.

1.1. Glass identification dataset

This dataset has information about different materials that appear in a glass, and the classification task is to decide to which category it belongs.

As mentioned before, this dataset is categorized as **small**, with 214 instances, each one with 10 attributes and a class.

A resume of the dataset can be seen in Table 1, which in the left part we can see the attribute index (columns) with the meaning, to later understand the feature importance, and the class attribute with its numerical labelling.

Attribute Index	**Attribute meaning**	**Class attribute**	
0	Id number	**1**	building_windows_float_processed
1	RI: refractive index	**2**	building_windows_non_float_processed
2	Na: Sodium	**3**	vehicle_windows_float_processed

3	Mg: Magnesium	**4**	vehicle_windows_non_float_processed
4	Al: Aluminium	**5**	containers
5	Si: Silicon	**6**	tableware
6	K: Potassium	**7**	Headlamps
7	Ca: Calcium		
8	Ba: Barium		
9	Fe: Iron		

Table 1. Glass dataset summarization

***Attributes from 3 to 10 are weight percent in corresponding oxide.*

1.2. Contraceptive Method Choice dataset

This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.

This dataset is categorized as **medium**, with 1473 instances, each one with 9 attributes and a class. A summarization of the dataset can be seen in Table 2.

Attribute index	Attribute names	Attribute meanings
0	Wife's age	(numerical)
1	Wife's education	(categorical) 1=low, 2, 3, 4=high
2	Husband's education	(categorical) 1=low, 2, 3, 4=high
3	Number of children ever born	(numerical)
4	Wife's religion	(binary) 0=Non-Islam, 1=Islam
5	Wife's now working?	(binary) 0=Yes, 1=No
6	Husband's occupation	(categorical) 1, 2, 3, 4
7	Standard-of-living index	(categorical) 1=low, 2, 3, 4=high
8	Media exposure	(binary) 0=Good, 1=Not good
9	Contraceptive method used	(class attribute) 1=No-use, 2=Long-term, 3=Short-term

Table 2. CMC dataset summarization

1.3. Chess (King-Rook vs. King-Pawn) dataset

This dataset has information over a chess game, where only a king-rook and a king-pawn are left in the board. A pawn on a7 is one square away from queening. The task is to determine if White can win or not (**won**, **nowin**), so we are facing a binary classification problem.

This dataset is categorized as **large**, with 3196 instances each one with 36 attributes and a class. The attributes represent a board position in the following way:

Attribute index	Attribute name	Attribute Description
0	bkblk	the BK is not in the way
1	bknwy	the BK is not in the BR's way
2	bkon8	the BK is on rank 8 in a position to aid the BR
3	bkona	the BK is on file A in a position to aid the BR
4	bkspr	the BK can support the BR
5	bkxbq	the BK is not attacked in some way by the promoted WP
6	bkxcr	the BK can attack the critical square (b7)
7	bkxwp	the BK can attack the WP
8	blxwp	B attacks the WP (BR in direction x = -1 only)
9	bxqsq	one or more Black pieces control the queening square
10	cntxt	the WK is on an edge and not on a8
11	dsopp	the kings are in normal opposition
12	dwipd	the WK distance to intersect point is too great
13	hdchk	there is a good delay because there is a hidden check
14	katri	the BK controls the intersect point
15	mulch	B can renew the check to good advantage
16	qxmsq	the mating square is attacked in some way by the promoted WP
17	r2ar8	the BR does not have safe access to file A or rank 8
18	reskd	the WK can be reskewered via a delayed skewer
19	reskr	the BR alone can renew the skewer threat
20	rimmx	the BR can be captured safely
21	rkxwp	the BR bears on the WP (direction x = -1 only)
22	rxmsq	the BR attacks a mating square safely
23	simpl	a very simple pattern applies
24	skach	the WK can be skewered after one or more checks
25	skewr	there is a potential skewer as opposed to fork
26	skrxp	the BR can achieve a skewer or the BK attacks the WP
27	spcop	there is a special opposition pattern present
28	stlmt	the WK is in stalemate
29	thrsk	there is a skewer threat lurking
30	wkcti	the WK cannot control the intersect point
31	wkna8	the WK is on square a8
32	wknck	the WK is in check
33	wkovl	the WK is overloaded
34	wkpos	the WK is in a potential skewer position
35	wtoeg	the WK is one away from the relevant edge

Table 3. Explanation of attributes in KRVSKP dataset

2. Project Architecture & pseudocode

There are different classes for this project. The following explanations will be from **deeper** to **higher** level. Also, files like main.py or utils.py will not be explained here, since they have not a direct impact in the classifiers, and more information can be found in the code documentation, inside the correspondent files and methods.

Class Node

This class is only implemented to save information about a node, and references to his left and right son. If it is a leaf, will have information in leaf_results parameter, or None otherwise. Also, it will also save the best attribute used in that node to split, and which attribute will send data to the right.

Class Cart

This is the base learner used in all the classifiers. It has two main (public) methods, fit and classify. By default, it uses all features in every node. If a number of features are passed, it will use that number of features when calculating impurity. This is used when the random forest calls CART and left as default in the Decision Forest.

Fit is the method where the dataset will be passed in to learn. It calls a recursive method (**_recursive_classification**) that works as follows:

```
If dataset = 0; return
Select F attributes
Calculate current impurity
If impurity != 0:
    For each attribute:
        For each attribute combination:
            Calculate and save lowest impurity
            Set keep expanding
If keep expanding:
    Expand left
    Expand right
    Return Node(left, right)
Else:
    Return Node(None, None, leaf_results)
```

Classify will return a list of classes of a given dataset to classify. It calls a recursive method (**_recursive_classification**) that work as follows:

For each row
- Recursive classification
  - If node is leaf return results
  - Else keep classifying to left or right
- Select final class
- Append class to results

Forest Classifier, Random Forest and Decision Forest

The architecture followed for this, was to create an abstraction (Forest Classifier), where the methods for predict, extract features and update features are implemented, since the only thing that it changes between Random Forest and Decision Forest is the fit method. With this, Random Forest and Decision forest **inherit** from Forest Classifier.

These methods work as follow:

Predict:

For each tree in the forest
- It performs classification

For each instance in the dataset
- We get the majority class between all trees

All classifications are returned

Features are saved when fitting, updated every time a tree in the forest is fit. Extract_features only returns the information saved in the forest.

The **fit** method is slightly different. Regarding **Random Forest**:

For each tree
- We subsamble the dataset by rows
- We fit the CART tree, adding the information of F parameters

For **Decision Forest:**

For each tree

If F is runif, we select a random F

Else we use the value given

We subsample the dataset by columns, depending on F

We fit the CART tree

3. Evaluation of results

In the following sections, results of different experiments will be presented. In order to increase readability, only the **four** most important features were put in the tables. To see the full list, the log of every experiment was saved in the **logs** folder of the project.

The best result is highlighted in **black**, taken into account by order of higher accuracy, then the smaller number of trees and finally the less parameters.

When talking about **feature importance**, the meaning of the rows is:

-('index of attribute', number of times that attribute appears in the forest), ex -> ('7', 107)

For **all** the 3D plots, if **runif** was among the F values, it was not considered.

3.1. Glass dataset with Decision Forest

The results of the glass dataset can be seen in Table 4.

Method	NT	F	Accuracy	Feature Importance
DecisionForestClassifier	1	2	0.963	('0', 5), ('2', 0)
DecisionForestClassifier	1	5	0.963	('0', 5), ('2', 0), ('6', 0), ('9', 0)
DecisionForestClassifier	1	7	0.963	('0', 5), ('2', 0), ('6', 0), ('9', 0)
DecisionForestClassifier	1	runif	0.963	('0', 4), ('6', 1), ('9', 0), ('7', 0)
DecisionForestClassifier	10	2	0.944	('1', 84), ('7', 71), ('6', 66), ('4', 57)
DecisionForestClassifier	10	5	0.963	('1', 65), ('7', 43), ('0', 27), ('6', 25)
DecisionForestClassifier	10	7	0.963	('0', 39), ('7', 12), ('6', 8), ('1', 8)
DecisionForestClassifier	**10**	**runif**	**0.981**	**('4', 112), ('5', 70), ('6', 53), ('7', 38)**
DecisionForestClassifier	25	2	0.907	('1', 216), ('6', 190), ('7', 185), ('4', 148)
DecisionForestClassifier	25	5	0.944	('7', 107), ('2', 101), ('1', 101), ('6', 78)
DecisionForestClassifier	25	7	0.963	('0', 82), ('7', 41), ('6', 33), ('3', 32)
DecisionForestClassifier	25	runif	0.963	('7', 223), ('4', 160), ('6', 114), ('3', 108)
DecisionForestClassifier	50	2	0.926	('4', 510), ('3', 449), ('7', 409), ('1', 346)
DecisionForestClassifier	50	5	0.963	('1', 197), ('2', 169), ('4', 152), ('7', 132)
DecisionForestClassifier	50	7	0.963	('0', 161), ('4', 78), ('7', 73), ('3', 66)
DecisionForestClassifier	50	runif	0.963	('4', 384), ('7', 258), ('2', 217), ('6', 207)

DecisionForestClassifier	75	2	0.907	('7', 735), ('4', 714), ('5', 590), ('1', 511)
DecisionForestClassifier	75	5	0.963	('1', 270), ('7', 256), ('4', 241), ('2', 212)
DecisionForestClassifier	75	7	0.963	('0', 245), ('7', 129), ('4', 100), ('3', 92)
DecisionForestClassifier	75	runif	0.963	('4', 585), ('2', 350), ('7', 347), ('6', 264)
DecisionForestClassifier	100	2	0.889	('7', 999), ('4', 858), ('5', 738), ('3', 665)
DecisionForestClassifier	100	5	0.963	('7', 345), ('4', 319), ('1', 312), ('6', 260)
DecisionForestClassifier	100	7	0.963	('0', 318), ('7', 175), ('4', 155), ('3', 124)
DecisionForestClassifier	100	runif	0.963	('4', 642), ('7', 541), ('2', 386), ('6', 334)

Table 4. Glass dataset results with Decision Forest Classifier

In this case, the behaviour of the model is clearer in the plot presented in Figure 1.

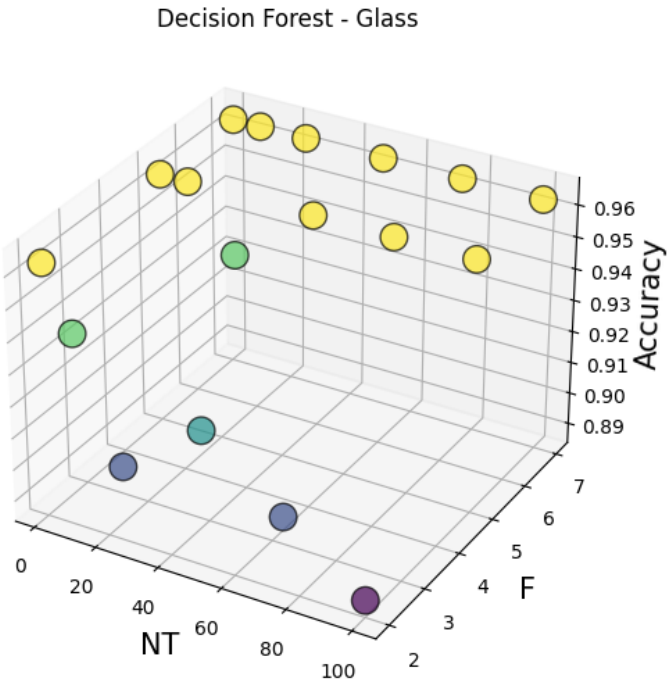

Figure 1. Decision Forest behaviour with Glass dataset

In this case we can see that the best results are obtained with more selected parameters. It looks like the number of trees does not have a high impact on the results. The nature of this results can be given by the size of the dataset. Being small, and using it entirely for the Decision Forest, makes it that the important thing is the number of features, not the number of trees, as we can confirm in the previous figure.

Regarding feature importance, it looks like the id_number (index 0), Calcium (index 7) and Aluminium (index 4) are the most important ones.

This is interesting because the id_number is **unique** for each instance. Calcium also makes sense since Limestone (Calcium carbonate) is one of the main components in glass manufacturing. Its main function is to introduce Calcium Oxide into the glass recipe, which is needed to improve chemical resistance and durability, hence giving an important clue of the final class.

3.2. Glass dataset with Random Forest

The results of the glass dataset with random forest can be seen in Table 5. The parameters of random features (**F**) were selected by hand, since the formulas with this dataset returned [1, 3, 3, 3] values. So, to avoid the same results, values of [1, 2, 3, 4] were selected.

Method	NT	F	Accuracy	Feature Importance
RandomForestClassifier	1	1	0.833	('0', 4), ('2', 3), ('4', 3), ('3', 2)
RandomForestClassifier	1	2	0.926	('0', 4), ('2', 1), ('3', 1), ('4', 1)
RandomForestClassifier	1	3	0.815	('0', 3), ('2', 3), ('3', 2), ('7', 2)
RandomForestClassifier	1	4	0.889	('0', 6), ('1', 1), ('2', 1), ('3', 1)
RandomForestClassifier	10	1	0.889	('0', 48), ('9', 41), ('7', 36), ('4', 35)
RandomForestClassifier	10	2	0.926	('0', 40), ('2', 31), ('4', 29), ('3', 25)
RandomForestClassifier	10	3	0.944	('0', 44), ('7', 16), ('2', 15), ('6', 13)
RandomForestClassifier	10	4	0.944	('0', 39), ('2', 8), ('3', 8), ('6', 8)
RandomForestClassifier	25	1	0.926	('0', 105), ('5', 85), ('2', 83), ('9', 82)
RandomForestClassifier	25	2	0.926	('0', 96), ('4', 88), ('1', 73), ('2', 73)
RandomForestClassifier	25	3	0.944	('0', 100), ('2', 45), ('7', 40), ('4', 39)
RandomForestClassifier	25	4	0.963	('0', 107), ('3', 32), ('1', 28), ('4', 23)
RandomForestClassifier	50	1	0.907	('0', 181), ('2', 161), ('1', 157), ('4', 156)
RandomForestClassifier	50	2	0.963	('0', 205), ('4', 149), ('2', 141), ('1', 134)
RandomForestClassifier	50	3	0.944	('0', 209), ('2', 101), ('7', 99), ('4', 83)
RandomForestClassifier	50	4	0.963	('0', 216), ('3', 67), ('1', 58), ('4', 58)
RandomForestClassifier	75	1	0.907	('0', 258), ('2', 256), ('4', 253), ('1', 249)
RandomForestClassifier	75	2	0.963	('0', 307), ('7', 207), ('3', 202), ('4', 201)
RandomForestClassifier	75	3	0.963	('0', 323), ('7', 147), ('2', 140), ('1', 133)
RandomForestClassifier	**75**	**4**	**0.981**	**('0', 328), ('3', 93), ('4', 83), ('1', 77)**
RandomForestClassifier	100	1	0.907	('4', 354), ('0', 345), ('1', 328), ('7', 327)
RandomForestClassifier	100	2	0.963	('0', 407), ('7', 286), ('4', 275), ('3', 269)
RandomForestClassifier	100	3	0.963	('0', 429), ('7', 191), ('2', 189), ('1', 178)
RandomForestClassifier	100	4	0.981	('0', 444), ('3', 121), ('4', 115), ('2', 107)

Table 5. Glass dataset results with Random Forest Classifier

A better behaviour of the classifier can be seen in Figure 2. As expected, we can see a tendency in better accuracy when the number of trees is high, and also when the number of features is higher.

Still, all results were good enough with more than 80% of accuracy with only one tree, as this problem is fairly simple.

Another thing that we can also observe is that there is a consensus between almost all experiments, and it looks like the most important feature is the id_number (index 0). It is clear that for almost every case, every tree will probably start splitting the nodes with this attribute, since it will be the one producing less impurity.

Then, for the higher number of trees, the second most important parameter is the Calcium (index 7), as in the previous classifier.

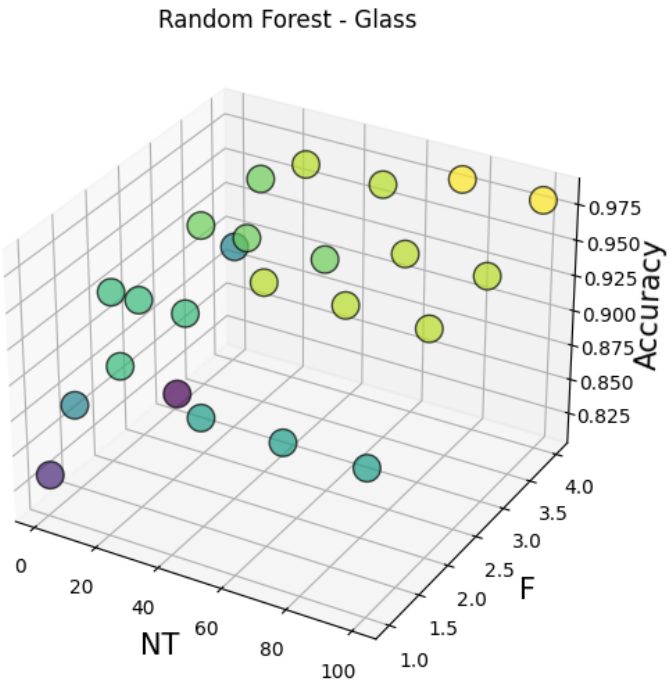

Figure 2. Random Forest behaviour with Glass dataset

3.3. CMC dataset with Decision Forest

In Table 7 we can observe the results of the Decision Forest with the CMC dataset.

Method	NT	F	Accuracy	Feature Importance
DecisionForestClassifier	1	2	0.375	('0', 83), ('2', 24)
DecisionForestClassifier	1	4	0.389	('0', 213), ('6', 84), ('2', 66), ('7', 51)
DecisionForestClassifier	1	6	0.405	('0', 223), ('7', 92), ('6', 87), ('2', 67)
DecisionForestClassifier	1	runif	0.473	('0', 184), ('3', 83), ('7', 78), ('6', 77)
DecisionForestClassifier	10	2	0.418	('0', 179), ('3', 86), ('7', 84), ('6', 28
DecisionForestClassifier	10	4	0.438	('7', 363), ('0', 348), ('6', 308), ('2', 285)
DecisionForestClassifier	**10**	**6**	**0.535**	**('0', 1659), ('7', 722), ('3', 640), ('6', 595)**
DecisionForestClassifier	10	runif	0.459	('0', 1074), ('3', 535), ('7', 350), ('6', 311)
DecisionForestClassifier	25	2	0.424	('0', 625), ('3', 156), ('7', 119), ('6', 58)
DecisionForestClassifier	25	4	0.459	('0', 1408), ('7', 838), ('6', 623), ('3', 599)
DecisionForestClassifier	25	6	0.508	('0', 3845), ('3', 1602), ('7', 1515), ('6', 1337)
DecisionForestClassifier	25	runif	0.470	('0', 2235), ('3', 934), ('7', 804), ('6', 664)
DecisionForestClassifier	50	2	0.435	('0', 1099), ('3', 283), ('7', 200), ('2', 101)
DecisionForestClassifier	50	4	0.465	('0', 3284), ('7', 1380), ('3', 1214), ('6', 1066)
DecisionForestClassifier	50	6	0.508	('0', 7822), ('3', 2741), ('7', 2713), ('6', 2539)
DecisionForestClassifier	50	runif	0.467	('0', 5177), ('3', 1971), ('6', 1591), ('7', 1510)

DecisionForestClassifier	75	2	0.432	('0', 1680), ('3', 572), ('7', 231), ('6', 136)
DecisionForestClassifier	75	4	0.454	('0', 5878), ('3', 2006), ('7', 1730), ('6', 1430)
DecisionForestClassifier	75	6	0.503	('0', 11941), ('3', 4164), ('7', 3801), ('6', 3700)
DecisionForestClassifier	75	runif	0.481	('0', 7581), ('3', 2919), ('6', 2477), ('7', 2317)
DecisionForestClassifier	100	2	0.424	('0', 2172), ('3', 710), ('7', 278), ('2', 166)
DecisionForestClassifier	100	4	0.462	('0', 8484), ('3', 2985), ('7', 2169), ('2', 1968)
DecisionForestClassifier	100	6	0.500	('0', 16246), ('3', 5740), ('6', 4876), ('7', 4766
DecisionForestClassifier	100	runif	0.476	('0', 10330), ('3', 4101), ('6', 3358), ('7', 3217)

Table 6. Decision Forest with CMC dataset

In this case, the results are not as good as the previous dataset. All results in terms of accuracy are almost the same, excluding those with only 1 tree. Still, we can see that the best results are with the most features. In Figure 3 we can perfectly see this improvement in accuracy when F is higher.

Where we can see some interesting information is in the feature importance. The most important feature is the Wife's age (index 0), followed by the number of children (index 3) and the standard of living (index 7). This totally makes sense, since this are the main factors to take into account when contraceptive methods are prescribed.

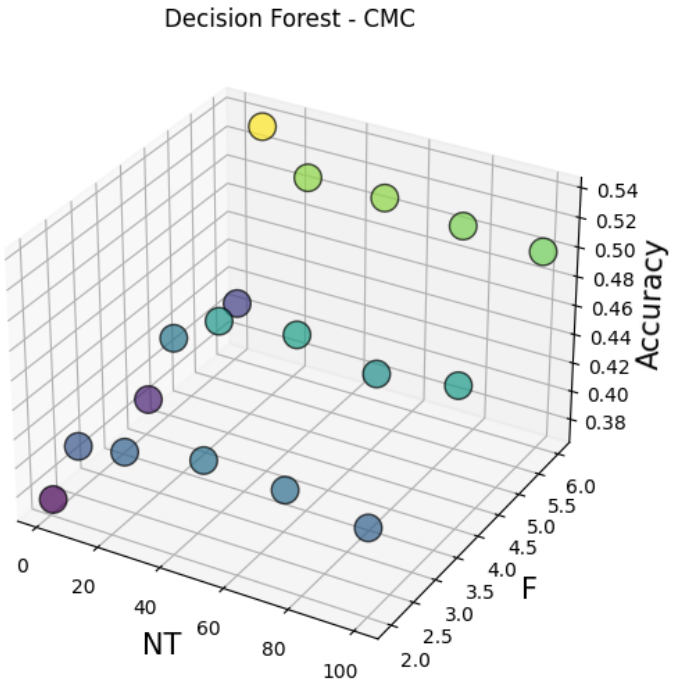

Figure 3. Decision Forest behaviour with CMC dataset

3.4. CMC dataset with Random Forest

In Table 7 we can see the results of the Random Forest Classifier with the CMC dataset.

Method	NT	F	Accuracy	Feature Importance
RandomForestClassifier	1	1	0.446	('0', 8), ('2', 6), ('3', 4), ('5', 3), ('6', 3)
RandomForestClassifier	1	2	0.418	('0', 57), ('3', 42), ('7', 28), ('2', 27)
RandomForestClassifier	1	3	0.421	('0', 93), ('3', 66), ('6', 43), ('7', 43)
RandomForestClassifier	1	4	0.462	('0', 116), ('3', 55), ('7', 47), ('6', 39)
RandomForestClassifier	10	1	0.486	('3', 76), ('0', 73), ('6', 58), ('1', 57)
RandomForestClassifier	10	2	0.508	('0', 483), ('3', 366), ('7', 253), ('6', 230)
RandomForestClassifier	10	3	0.473	('0', 844), ('3', 594), ('7', 381), ('6', 364)
RandomForestClassifier	10	4	0.486	('0', 1084), ('3', 636), ('7', 473), ('6', 342)
RandomForestClassifier	25	1	0.497	('0', 235), ('3', 222), ('1', 186), ('2', 170)
RandomForestClassifier	25	2	0.508	('0', 1258), ('3', 953), ('7', 637), ('6', 581)
RandomForestClassifier	25	3	0.503	('0', 2128), ('3', 1531), ('7', 926), ('6', 869)
RandomForestClassifier	25	4	0.489	('0', 2747), ('3', 1598), ('7', 1164), ('6', 906)
RandomForestClassifier	50	1	0.516	('0', 447), ('3', 403), ('1', 329), ('6', 316)
RandomForestClassifier	**50**	**2**	**0.535**	**('0', 2514), ('3', 1963), ('7', 1265), ('6', 1194)**
RandomForestClassifier	50	3	0.505	('0', 4214), ('3', 2983), ('7', 1911), ('6', 1737)
RandomForestClassifier	50	4	0.508	('0', 5488), ('3', 3166), ('7', 2310), ('6', 1813)
RandomForestClassifier	75	1	0.495	('0', 666), ('3', 617), ('1', 496), ('2', 459)
RandomForestClassifier	75	2	0.522	('0', 3799), ('3', 2951), ('7', 1885), ('6', 1779)
RandomForestClassifier	75	3	0.514	('0', 6448), ('3', 4485), ('7', 2832), ('6', 2613)
RandomForestClassifier	75	4	0.514	('0', 8188), ('3', 4863), ('7', 3412), ('6', 2725)
RandomForestClassifier	100	1	0.503	('0', 845), ('3', 778), ('1', 620), ('2', 613)
RandomForestClassifier	100	2	0.527	('0', 5125), ('3', 3968), ('7', 2493), ('6', 2384)
RandomForestClassifier	100	3	0.519	('0', 8648), ('3', 5992), ('7', 3782), ('6', 3484)
RandomForestClassifier	100	4	0.519	('0', 10937), ('3', 6470), ('7', 4444), ('6', 3751)

Table 7. Random Forest with CMC dataset

As with the previous classifier, the results in terms of accuracy do not vary a lot. We can again see the confirmation that the most important features are the age, the number of children, and finally the standard of living, which makes total sense so far.

In Figure 4 we can see a little bit clearer, that in this case the greater number of features and the greater number of trees, the better results.

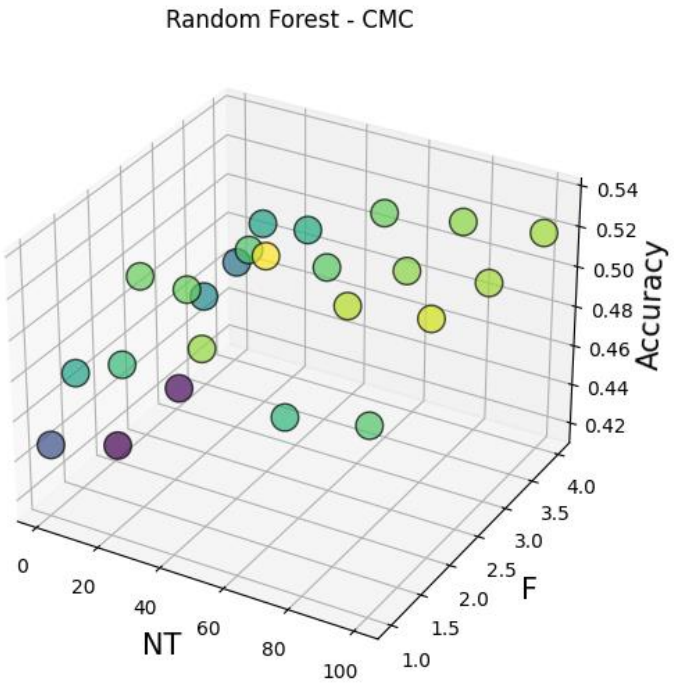

Figure 4. Random Forest behaviour with CMC dataset

3.5. KR-vs-KP dataset with Decision Forest

In Table 8 we can see the results of the Decision Forest Classifier with the KRVSKP dataset.

Method	NT	F	Accuracy	Feature Importance
DecisionForestClassifier	1	9	0.625	('3', 15), ('8', 12), ('11', 12), ('25', 7)
DecisionForestClassifier	1	18	0.849	('23', 43), ('5', 31), ('8', 28), ('11', 25)
DecisionForestClassifier	1	27	0.960	('23', 15), ('17', 12), ('5', 11), ('8', 9)
DecisionForestClassifier	1	runif	0.954	('17', 11), ('21', 8), ('29', 7), ('6', 6)
DecisionForestClassifier	10	9	0.824	('23', 51), ('25', 49), ('8', 36), ('19', 35)
DecisionForestClassifier	10	18	0.925	('23', 249), ('11', 152), ('35', 148), ('25', 143)
DecisionForestClassifier	10	27	0.981	('23', 355), ('35', 214), ('5', 205), ('4', 190)
DecisionForestClassifier	10	runif	0.914	('23', 145), ('5', 106), ('8', 95), ('35', 80)
DecisionForestClassifier	25	9	0.869	('11', 105), ('4', 103), ('23', 102), ('29', 75)
DecisionForestClassifier	25	18	0.942	('23', 547), ('4', 370), ('11', 363), ('25', 290)
DecisionForestClassifier	25	27	0.986	('23', 732), ('4', 441), ('25', 406), ('5', 398)
DecisionForestClassifier	25	runif	0.879	('23', 364), ('12', 240), ('25', 224), ('35', 205)
DecisionForestClassifier	50	9	0.891	('23', 190), ('4', 170), ('11', 169), ('19', 142)

DecisionForestClassifier	50	18	0.966	('23', 842), ('4', 629), ('11', 626), ('17', 580)
DecisionForestClassifier	50	27	0.989	('23', 1307), ('4', 786), ('12', 714), ('5', 706)
DecisionForestClassifier	50	runif	0.941	('23', 800), ('5', 530), ('12', 464), ('11', 449)
DecisionForestClassifier	75	9	0.894	('23', 300), ('11', 247), ('19', 229), ('4', 229)
DecisionForestClassifier	75	18	0.971	('23', 1228), ('4', 1016), ('11', 917), ('12', 830)
DecisionForestClassifier	**75**	**27**	**0.994**	**('23', 2006),('4', 1169),('12',1126),('25',1017)**
DecisionForestClassifier	75	runif	0.974	('23', 963), ('12', 631), ('5', 626), ('11', 592)
DecisionForestClassifier	100	9	0.881	('23', 339), ('11', 317), ('19', 300), ('4', 292)
DecisionForestClassifier	100	18	0.966	('23', 1537),('4', 1374),('12', 1208),('11', 1136)
DecisionForestClassifier	100	27	0.994	('23', 2538), ('4', 1550), ('12', 1468), ('5', 1335)
DecisionForestClassifier	100	runif	0.974	('23', 1257), ('12', 989), ('25', 822), ('5', 816)

Table 8. Decision Forest with KRVSKP dataset

With this dataset, the results are again great. And the behaviour of the classifier is also very smooth to interpret, as we can also see in Figure 5.

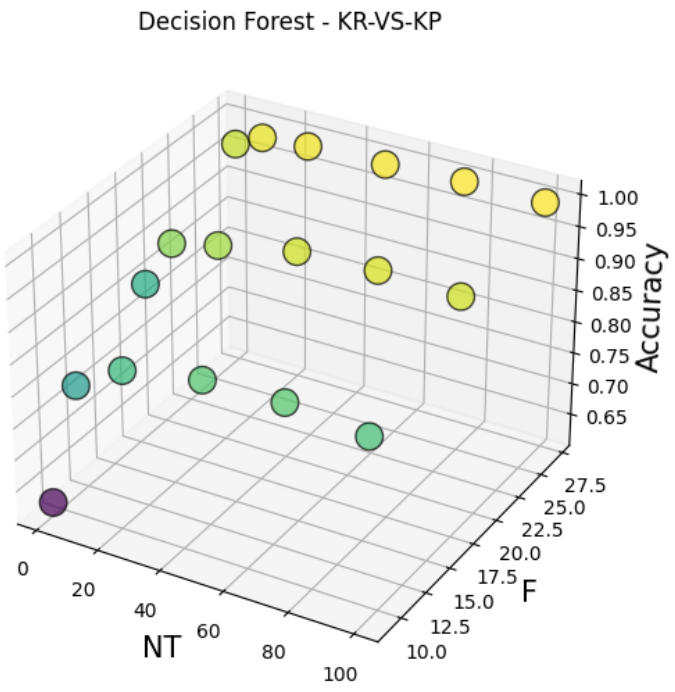

Figure 5. Decision Forest behaviour with krvskp dataset

Basically, the higher the number of parameters and trees, the better accuracy we have. Actually, we can see a nice increasing surface in the previous Figure.

Taking a look at the feature importance, we can see that the most important one is simpl (index 23), followed by bkspr (index 4).

Again, this feature makes sense, cause if by using a simple pattern we can win or lose the game (simpl), or if the black king supports the black rook, which they are already in advantage because they are playing vs king-pawn, is a very decisive position to win the game.

3.6. KR-vs-KP dataset with Random Forest

In Table 9 and in Figure 6 we can see the results of the Random Forest Classifier with the KRVSKP dataset.

Method	NT	F	Accuracy	Feature Importance
RandomForestClassifier	1	1	0.681	('9', 3), ('11', 3), ('10', 2), ('14', 2)
RandomForestClassifier	1	3	0.819	('5', 4), ('10', 4), ('14', 4), ('20', 4)
RandomForestClassifier	1	5	0.922	('32', 11), ('23', 9), ('5', 8), ('12', 7)
RandomForestClassifier	1	6	0.965	('5', 8), ('17', 7), ('21', 7), ('33', 7)
RandomForestClassifier	10	1	0.727	('20', 11), ('30', 11), ('29', 9), ('6', 8)
RandomForestClassifier	10	3	0.946	('14', 65), ('5', 58), ('32', 58), ('20', 56)
RandomForestClassifier	10	5	0.979	('32', 89), ('9', 79), ('5', 78), ('14', 73)
RandomForestClassifier	10	6	0.985	('32', 78), ('6', 74), ('5', 73), ('33', 71)
RandomForestClassifier	25	1	0.711	('14', 22), ('30', 20), ('1', 19), ('3', 19)
RandomForestClassifier	25	3	0.970	('9', 155), ('20', 150), ('5', 145), ('14', 145)
RandomForestClassifier	25	5	0.985	('5', 229), ('20', 211), ('23', 203)
RandomForestClassifier	25	6	0.989	('5', 223), ('20', 208), ('23', 204), ('32', 196)
RandomForestClassifier	50	1	0.756	('14', 52), ('0', 39), ('20', 39), ('10', 37)
RandomForestClassifier	50	3	0.970	('20', 329), ('14', 307), ('9', 303) , ('5', 295)
RandomForestClassifier	50	5	0.987	('20', 432), ('32', 415), ('23', 414), ('5', 408)
RandomForestClassifier	50	6	0.992	('5', 455), ('20', 445), ('23', 436), ('32', 387)
RandomForestClassifier	75	1	0.746	('14', 74), ('0', 55), ('20', 55), ('9', 54)
RandomForestClassifier	75	3	0.966	('20', 522), ('14', 457), ('9', 444), ('32', 442)
RandomForestClassifier	75	5	0.989	('32', 668), ('20', 650), ('23', 618), ('5', 603)
RandomForestClassifier	**75**	**6**	**0.995**	**('5', 676), ('20', 676), ('23', 644), ('32', 609)**
RandomForestClassifier	100	1	0.747	('14', 104), ('20', 81), ('0', 80), ('10', 76)
RandomForestClassifier	100	3	0.969	('20', 686), ('32', 586), ('9', 577), ('14', 570)
RandomForestClassifier	100	5	0.992	('20', 873), ('32', 844), ('23', 806), ('5', 805)
RandomForestClassifier	100	6	0.995	('20', 887), ('5', 882), ('32', 846), ('23', 840)

Table 9. Random Forest with KRVSKP dataset

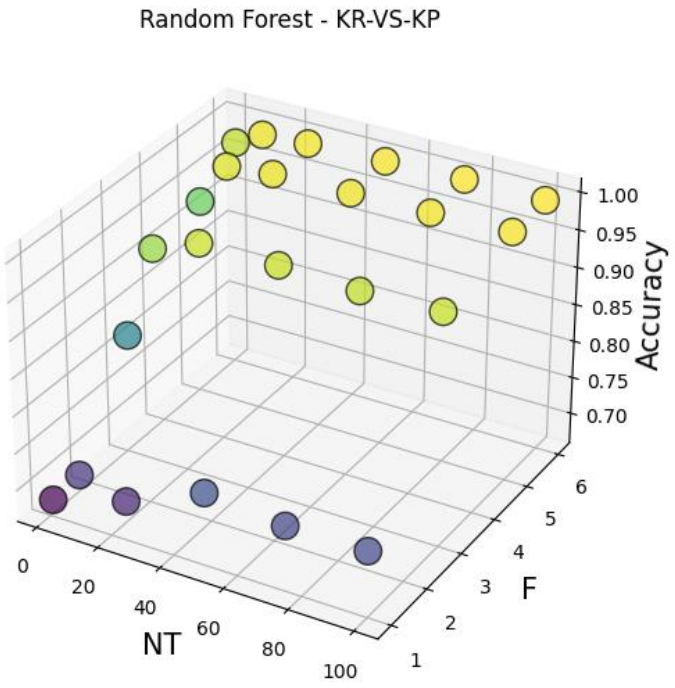

Figure 6. Random Forest behaviour with krvskp dataset

The better results are very similar to the Decision Forest in terms of best accuracies, reaching an almost 100%. In this case thought, a low number of parameters will give worse results, it does not matter how many trees we have, as we can see in Figure 6.

In this case the feature importance is not so clear. If we take the best results in terms of accuracy, the most important feature is rimmx(index 20), which means that the BR can be captured safely. This is a very clear indication that the whites can win.

4. Conclusions

In conclusion, the algorithms implemented give nice results on all the experiments, with an understandable feature importance, and results that makes sense with the problem being tackled. Also, the results are very similar.

What it can be seen is that in random forest, the number of features must go paired with the number of trees. This means that if we have a low number of features, the results will not drastically change, it does not matter how many trees we use. This is something that with Decision Forest does not happen as clear as in Random Forest.

Another point is that Decision Forest needs much more parameters compared to Random Forest, to give the same results. This behaviour can be specially seen in KRVSKP test, where with only 6 parameters we already have almost a perfect accuracy, meanwhile with Decision forest we need almost 24 parameters to obtain the same. This behaviour also makes sense, because despite in random forest we

are using a bootstrapped sample of the original dataset, we are using almost all features in the tree, because random F features are being selected per node. In decision forest, we are limited to F features per tree. So, it does not matter how many instances do we send to the tree, if the parameters are not good, the tree will not be very useful.

To summarize, we can see that random forest are very powerful supervised learning methods. Another advantage that they have is that they can be easily parallelized due to the nature of the algorithm. Apart from this, they are not very computationally demanding, and the execution times can be reduced thanks to different pruning techniques.

How to execute

The main.py is the file that needs to be executed in order to evaluate a classifier. This file is configured with a json file, that needs to be given as parameter.

The structure of the json is the following:

```
{
        "classifier": str,
        "dataset": str,
        "train_test_split": float,
        "NT": List[int],
        "F": List[int],
        "csv_out_name": str,
        "seed": integer,
}
```

All parameters but **"F"** are required. The explanation of the parameters is the following:

- Classifier: Name of the Classifier that will be used by the forest interpreter. In this case, **only RandomForestClassifier** or **DecisionForestClassifier** are **valid** parameters.
- Dataset: Path to the dataset that will be used in the experiment. Remember that the path is relative to your location when running the main.py file. In this case, you probably have to go one folder behind, since the datasets are saved in Data.
- Train_test_split: Fraction of the test data used for train; the rest will be used for test.
- NT: Number of trees that the forest interpreter will use.
- F: Number of parameters that the forest interpreter will use. This is *optional*, since the default number of parameters is explained in Introductions and selected datasets. This field was introduced because two of the datasets had a low number of features, hence repeating the values sometimes.
- Csv_out_name: Name of the .csv and info files where the main results (Method, number of trees, number of parameters and accuracy, feature importance and accuracy respectively) will be saved. This file will be saved in source/logs and will **overwrite** any existent file with the same name. The name of the csv will be in format logs/{csv_out_name}, and the information file will be saved as logs/{csv_out_name}_NT-{NT}_F-{F}.
- Seed: Seed used to obtain reproducible results.

To summarize, to run the code you only need to do: **python main.py json_file**

A **README.md** file with detailed information of the project structure and examples of how to run the code can be found in the project.

References

[1] T. K. Ho, “The Random Subspace Method for Constructing Decision Forests.,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 20(8):832-844, 1998.

[2] L. Breiman, “Random Forests,” *Machine Learning*, pp. 45:5-32, 2001.

[3] Dua, Dheero; Graff, Casey, “UCI Machine Learning Repository,” 2019. [Online]. Available: http://archive.ics.uci.edu/ml. [Accessed 14 5 2021].

Figures

Tables